

Crônica: O Maior Software já Escrito em Todos os Tempos.

Autoria de Redação FUG-BR
16/08/2006
Última Atualização 18/08/2006

Charles Babcock um dos principais redatores da revista eletrônica sobre tecnologia da informação, Information Week disserta em uma crônica onde analisa algumas das principais tecnologias criadas pelo homem para tentar classificar: O Maior Software Escrito.

Babcock considera softwares pouco lembrados, como o Sabre da American Airlines, ou programas que sequer eram softwares, como o Colossus. Remete-se a clássicos como IBM 360 e Deep Blue, para finalmente concluir qual o maior programa de computador já escrito em todos os tempos. O autor define critérios claros para suas conclusões: "Desenvolvimento de fato superior só pode ser julgado dentro de um contexto histórico. Deve representar uma inovação, brilhantismo técnico, algo difícil que antes de tal software, não poderia ser conseguido. E deve ter plena adoção no mundo real. O Colossus transformou um processo mecânico de grafia em eletrônica -- foi uma espécie de computador pioneiro -- e ofereceu um serviço útil ao acelerar a tradução do telégrafo codificado. O Colossus mais que fez história, ajudou a história a tomar forma." Ao revisar as maiores e mais impactantes criações tecnológicas da história, o autor repassa pela criação dos sistemas operacionais, da interface gráfica com o usuário, planilhas eletrônicas, da Internet, do Unix, TCP/IP, DNS, correio eletrônico, editores de texto e conclui: "O Maior Software de Todos os Tempos, com o mais amplo impacto no mundo todo foi o BSD 4.3. Outros Unix tornaram-se sucesso comercial mais bem sucedido, mas as conquistas cumulativas conseguidas nos sistemas BSD, especialmente versão 4.3, representaram picos incomparáveis de inovação. O BSD 4.3, representa sozinho, a maior revolução da Internet. Mais que isso, a paixão que envolve o Linux e código aberto é apenas uma ramificação direta dos ideais criados no BSD: amor pelo poder computacional e a crença que este deve ser uma extensão do poder intelectual humano, disponível sob todos os preceitos de liberdade -- poder de intelecto, este que coloca o homem em seu devido lugar no universo." A tradução dessa crônica foi republicada internamente na FreeBSD Brasil LTDA, e essa tradução você acompanha na íntegra agora, aqui na FUG-BR.

O Maior Software já Escrito em Todos os Tempos

Por Charles Babcock, 14 de Agosto de 2006

Sem exceção, todo profissional de tecnologia pode opinar de forma rápida sobre qual o maior software já produzido, mas quando reserva-se um tempo para avaliar o que torna um software realmente brilhante, as escolhas não são tão óbvias. Uma das mais significantes criações da programação que conheço, não era sequer um software. Antes dos britânicos construírem a máquina Colossus, que traduziam os códigos de telégrafo dos alemães durante a Segunda Guerra Mundial, os aliados demandavam seis horas para decodificar uma mensagem, e mais de um dia para interpretar essas informações junto à inteligência, chegar a conclusões e finalmente repassar essas informações ao comando militar. Depois da Colossus, os aliados conseguiram decodificar essas informações imediatamente ao interceptá-las, e assim, ter um panorama visual de todas as atividades militares alemãs na Inglaterra -- informações usadas pelo General Dwight Eisenhower para, com segurança, lançar o ataque do Dia D. Colossus foi construída em 1944 para realizar operações Booleanas em formulário contínuo gerado na máquina na grandeza de 30 milhas de papel por hora. Sua lógica era literalmente integrada e dependente da máquina. É, talvez, o maior software que nunca foi escrito. Mas então, a que conclusão isso nos leva? Primeiro, vamos definir critérios para avaliar o que torna um software grandioso. Desenvolvimento de fato superior só pode ser julgado dentro de um contexto histórico. Deve representar uma inovação, brilhantismo técnico, algo difícil que antes de tal software não poderia ser conseguido. E deve ter plena adoção no mundo real. A Colossus transformou um processo mecânico de grafia em eletrônica -- foi uma espécie de computador pioneiro -- e ofereceu um serviço útil ao acelerar a tradução de telégrafo codificado. Colossus mais que fez história, ajudou a história a tomar forma. Outro exemplo de grande programação foi o sistema 360 da IBM. O software foi escrito como o primeiro sistema operacional de propósito geral, em 1964. Muitas das verdades que conhecemos hoje sobre software -- por exemplo, que projetos simples são melhores que os complicados, que poucos programadores habilidosos conseguirão melhores resultados do que pelotões de programadores igualmente habilidosos -- são originados do livro de Frederick Brooks sobre o projeto do 360, *The Mythical Man-Month*, da editora Addison-Wesley Professional de 1995. Brooks já sabia a quantidade de problemas que poderiam ocorrer com grandes projetos de software, antes que o projeto 360 tivesse início. De fato, ele era um crítico sobre como a IBM deveria gerir o projeto; ele considerava que os potenciais de falha eram muitos. Suponho que seja por isso que a IBM o tornou responsável pelo projeto. Muito esperto. O resultado foi o primeiro sistema computacional capaz de executar aplicações distintas ao mesmo tempo. Originou a linha de mainframes da IBM, que posteriormente evoluiu até a Série 370 e atualmente na zSeries. Até hoje, esses sistemas são compatíveis com o sistema operacional IBM 360 de Brooks. O que me faz lembrar de outro atributo de um grande software: tornar-se tradicional. Ficou e está há tanto tempo no mercado, que não pode ser facilmente substituído. Reconhecemos sua grandeza. Todos concordam que o IBM 360 foi um dos maiores softwares já escritos. Sua grandiosidade é facilmente avaliada dadas suas longas perspectivas históricas. Contudo, quanto mais perto ficamos do presente, mais difícil fica classificar os grandes softwares. Bem, ainda temos torpedos. Com grande inspiração o coloquei em minha lista de grandes softwares já escritos, partindo da Colossus até o presente. Consultei os gurus James Rumbaugh; Stuard Feldman, o presidente da Associação de Maquinários Computacionais, a investidora Ann Winblad e Gary Morgenthaler; os criadores da linguagem de scripting PHP 3.0, Zeev Suraski e Andi Gutmans; e meu pequeno irmão, Wally. Essa lista todavia, continua bastante pessoal. Os que a considerarem sábia e com escolhas inspiradas, me envie uma mensagem de correio eletrônico no endereço ao final dessa crônica. Aos que considerarem de mal gosto, sem fundamentos ou horrivelmente ignorante, enviem sua mensagem ai Wally, ex estrela profissional de basketball de

aproximadamente 2,10 metros de altura. Sempre fiquei impressionado com a nave espacial Apollo e seu sistema de orientação, criado pelo Laboratório de Instrumentação do MIT. Em 1969 esse software levou a Apollo 11 para a lua, desconectou o módulo lunar, pousou-a na superfície da lua e trouxe três astronautas de volta para casa. Tinha que funcionar na pequena quantidade de memória disponível no computador Raytheon que estava onboard na nave. O software tinha 8 Kbytes, menos que um driver de impressora hoje em dia. E não poderia haver a menor hipótese de reboot em caso de falha no sistema, quando a nave estivesse fazendo a re-entrada em atmosfera terrestre. Felizmente o Windows não estava realizando o serviço. O sistema de orientação da Apollo hoje pode parecer software de rotina, do ponto de vista da sofisticação tecnológica. Sistemas de navegação muito mais complexos são operados hoje em dia. A essência do sistema era um conjunto de algoritmos conhecidos baseados em lógica comprovada. Mas para mim, ainda é Ciência Espacial. Grandes softwares nos impressionam pela segurança com que desempenham corretamente tarefas que tem tudo para dar errado. Aos que não se impressionam pela relativa simplicidade do sistema espacial da Apollo, pergunto: Você colocaria sua vida sob dependência de um sistema mais complexo? Por exemplo, o Sistema de Automação BAE, software que supostamente deveria manusear as bagagens no Aeroporto Internacional de Denver, projetado em dois supercomputadores de processamento paralelo. Em sua data de lançamento, outubro de 1993, direcionou tanta bagagem para os aviões errados que a cidade teve que adiar a inauguração do aeroporto em 16 meses. O prejuízo consequente para a cidade foi equivalente a 1.1 milhão de dólares por dia. Apenas para ilustrar, nossas vidas já estão nas mãos desse tipo de software. A Administração da Aviação Federal nos EUA gastou centenas de milhões de dólares, não apenas uma, mas três vezes tentando construir um sistema efetivo de tráfego aéreo. Jogou fora cerca da metade do que criou, tecnologia avaliada em 144 milhões de dólares, enquanto a outra metade desse montante, a metade que se tornou o sistema em uso, frequentemente falha ou para completamente. Grande software? Dou preferência para o sistema de orientação da Apollo. Para que um software seja considerado um sucesso, o mínimo que se espera é que ele consiga realizar as tarefas à que foi criado. Certamente tal axioma se aplica ao VisiCalc, o primeiro software de planilha eletrônica. Ele é grande, pois demonstrou o potencial da computação pessoal. O software colocou nas mãos de cada empresa a possibilidade de analisar e manipular enormes quantias de dados matemáticos. Mas o VisiCalc em si, apesar de representar um conceito inovador, não era um grande software. Era falho e limitado, não podia desempenhar muitas das tarefas que os usuários desejavam. A grande implementação da planilha de cálculos não foi VisiCalc, e nem mesmo o Lotus 1-2-3, mas sim o Microsoft Excel, que estendeu o poder das planilhas eletrônicas e dispôs às empresas uma grande variedade de ferramentas de cálculo. A Microsoft afirma que faz software brilhante, sem dúvida é uma afirmação no mínimo contestável, mas a planilha eletrônica Excel veio para ficar! É unânime para todos que usam profissionalmente esse tipo de ferramenta.

Em Busca de Inteligência

Há muitos exemplos de grandes softwares no campo da inteligência artificial. Cada um ao seu tempo. Al por exemplo, produziria inteligência similar à humana, podendo inclusive conversar conosco, nos ensinar algo que não entendemos e combinar grande poder e racionalidade com comandos e dados sem um limite. Mas afinal, como acabou o AI? Muitos artificios e inteligência insuficiente. Redes neurais, conceito criado na pesquisa de desenvolvimento do AI, produziu sistemas de identificação de impressões digitais usados plenamente no mundo todo. É sem dúvida um grande sistema de combinação de padrões, mas isso é inteligência? Minha lógica diz que não. A aplicação AI que produziu a primeira inovação foi a máquina de inferência, sistema dotado de base de conhecimento e regras de condições. Esse sistema computacional é capaz de combinar condições que identificam precisamente um paciente com febre antes mesmo dele atingir o grau de calor corporal necessário para ser considerado febre, simplesmente fazendo uso de uma regra, o fato de infecções bacteriais causar grandes febres. Uma das melhores variantes desse software o sistema de diagnóstico médico Mycin, pode corretamente identificar infecções bacteriais em uma pessoa baseado em seus sintomas com precisão de 65%. Pouco? É melhor que a maioria dos profissionais recém graduados na mesma área. Mas esse tipo de software nunca saiu do laboratório e tampouco tornou-se popular. Ninguém poderia identificar o culpado quando seu resultado não fosse preciso. Meu pacote favorito do AI foi o Deep Blue da IBM, que derrotou o grande campeão Mundial de xadrez Garry Kasparov em um jogo de seis partidas. Kasparov alegou que por trás da parafernália eletrônica haviam humanos. E ele estava certo. Os programadores da IBM estavam furiosamente revisando as decisões do Deep Blue entre cada jogada e ajustando o computador de acordo com o estilo de jogo de Kasparov. Isso retira completamente o Deep Blue da minha lista de candidatos à grande software. As atividades da IBM condisseram com as regras da disputa, mas não deveria ter sido assim. Como Kasparov poderia competir? Readaptando seus circuitos mentais? Os softwares no estilo AI podem impressionar, mas todos os meus melhores exemplos ficam longe de ser considerados grandiosos. Eventualmente descrevo meu navegador de Internet como um terminal burro emocionalmente dotado. Meu irmão, Wally, pesquisador literário convenceu-me que o Mosaic foi o primeiro navegador gráfico, "tirou a Web da terra dos técnicos e a colocou no mundo dos humanos comuns". Um de seus predecessores, o Gopher, é um exemplo de quase sucesso, e depois veio o ViolaWWW, o primeiro navegador com os botões de páginas anteriores e próximas. Mas a combinação de linha de endereço para a URL, interface de apontar e clicar baseada em mouse, apresentação de arquivos multimídia e hyperlink na janela, significava que os clientes finalmente haviam encontrado uma parceria perfeita para acessar as informações que proliferavam nos servidores da Internet. A combinação de elementos de fácil uso do Mosaic com um conjunto de menus deslocáveis, criaram um formato que mais tarde seria sempre reproduzido pelo Netscape Navigator, pelo Internet Explorer e pelo Firefox (na janela do Explorer experiente clicar no menu Ajuda e depois no Sobre ou About, e você verá os créditos ao Mosaic). Brilhantismo técnico? Não exatamente, mas uma necessidade solene e síntese de novidade técnica. Em outras palavras, um grande software que abriu diversas portas. O mesmo pode ser dito da própria World Wide Web? Tim Berners-Lee produziu a síntese da linkagem de hypertextos, do localizador de recursos universal (URL), e as páginas HTML impactaram nosso mundo monstruosamente. Mas a Web copiou idéias que já existiam, e todas são dependentes dos protocolos de rede TCP/IP e do servidor de nomes BIND (Berkeley Internet Domain), aquela camada de software próxima do metal que faz por exemplo, roteadores funcionar. Não, a Web

não é um grande software, mas certamente é um dos líderes de escala quando consideramos a popularidade de seu impacto. Satisfação do Usuário Continuando a análise contemporânea, Google, em ao menos um aspecto, representa um grande software. Os sistemas de busca precediram o Google na forma do Lycos do AltaVista da DEC e outros engines. Mas o Google incorporou um sistema de ranking de páginas estruturadas em seus resultados de busca, associando as milhares de páginas resultantes de uma pesquisa em um sistema hierárquico que representa a frequência de seu uso ou de referências cruzadas em outras páginas. "O valor de um documento acadêmico é medido de acordo com o número de vezes que esse documento é mencionado em outros documentos ou em suas notas de rodapé". O Google adaptou essa convenção para a Web, segundo Morgenthaler Ventures. Além disso, colocou na mão de milhões de novos usuários uma grande e valiosa ferramenta de informação estruturada. Taí um grande software. Pensei também no Java da Sun, como linguagem derivativa, membro da família C que redefiniu convenções que já existiam. Depois de refletir, concluí que estava errado. Java implementa o conceito de máquina virtual em clientes, permitindo que o código fosse transportado pelas redes e executado no PC de destino sem que este código tenha muitas informações dessa máquina. Java instituiu o uso de byte code intermediário, uma espécie de código fonte previamente compilado, que permite sua tradução para código de máquina assim que chega na plataforma cliente. Isso equaciona portabilidade e performance. Java restringiu o código copiado para uma espécie de sandbox ou um conjunto de limites -- o disco rígido do cliente por exemplo, fica estritamente nesses limites. A sandbox em teoria protege os usuários de se expor à problemas de segurança, como os problemas encontrados com sistemas irrestritos como o ActiveX da Microsoft. Com esses recursos orientados à rede, Java deslizou no mundo dos negócios e na era da Internet. A Microsoft copiou todas as melhores idéias por trás do Java ao criar o Visual Studio .Net. Java se destacou sobre essa cópia, se estendeu e tornou-se mais rico em termos de recursos, e isso certamente é sinal de grandiosidade. E se pensarmos em software mais focado em usuários finais, como publicação baseada em Desktop? Esse tipo de adoção foi possível com o PostScript da Adobe Systems, que digitalmente pode formatar imagens e textos dentro de um computador ou em uma impressora laser. Adobe simplificou as definições tipográficas profissionais que surgiram nos laboratórios de pesquisa da Xerox em Palo Alto no Vale do Silício, conseguindo uma combinação coerente de simplicidade e poder operacional no formato PostScript. Tornou a publicação baseada em estações de trabalho convencionais totalmente comum. Muito bem feito, mas não o suficiente para ser considerado inovação técnica e chamado de grande software.

Falando na Xerox, o Macintosh da Apple foi baseado no sistema denominado Alto, criado na Xerox. Alto incluiu a primeira interface gráfica baseada em janelas, o primeiro mouse e a primeira interface unificada para o usuário. Mas nunca chegou no mercado. Foi necessário completo redesign do projeto pela Apple para gerar o impacto. Posso ainda me lembrar da primeira vez que sentei na frente de um Macintosh, na vitrine de uma loja de computadores em Endicott, N.Y. Eu tive aquela sensação de "Ciência Espacial": podia ver o que o sistema fazia, mas não conseguia acreditar no que via. O Mac incorporou o poder da computação orientada a objetos na interface com o usuário, e a maioria dos usuários nunca mais quiseram saber de outra coisa. O primeiro sistema operacional Mac foi um grande software.

É Chegado o Momento das Pestes

Tecnologia que se infiltra em nosso dia-a-dia e modifica completamente nossa vida se qualifica como grande software. De tal forma, meu próximo candidato consegue alcançar esse critério, mesmo sendo um pedaço desprezível de software. Em 1988 a peste Morris (um worm de rede) se proliferou pela Internet, infiltrando-se em servidores de grandes universidades e fechando grandes escritórios de negócios. O estudante de Cornell, Robert Morris, afirma hoje em dia que fez o work para que pudesse mensurar o tamanho da Internet apropriadamente.

Como a maioria dos software, o worm teoricamente poderia rodar em apenas um ou dois ambientes alvo, mas acabou demonstrando algo novo sobre a grande rede de computadores. A peste em questão se espalhou de servidor em servidor explorando uma vulnerabilidade de estouro de pilha no Sendmail. Não podíamos imaginar quantos backdoor ou aberturas defensivas puderam atordoar o Unix, Sendmail, Finger e outros sistemas. A peste continuamente definia novos servidores como alvo e, aleatoriamente, se replicava. Morris afirma ter adicionado esse recurso para garantir que seu worm pudesse se espalhar. Ele conseguiu.

Como software, esse intruso foi uma inovação, uma demonstração capaz de abrir os olhos dos mais céticos sobre como um software brilhante poderia explorar um lado anti-social até então não conhecido nessa proporção na Internet. Estávamos todos começando a ficar interconectados e dependentes disso. Precisávamos de um puxão de orelha, algo para nos acordar. Parabéns, Senhor Morris. A Grande Peste se provou um incontestável e preciso alarme. Era um grande software. O sistema Sabre da American Airlines também era grande. Demonstrou como software poderia ser usado além das necessidades táticas de um negócio, mas também como recurso estratégico. Sabre tinha a habilidade de calcular e combinar as necessidades de viajens de um cliente com os vôos disponíveis, em um escritório de um agente de viagens. Sua listagem incluía também vôos dos competidores da companhia aérea em questão. O sistema garantiu economia de tempo e dinheiro junto à American Airlines e também seus para seus agentes de viagens. Mas a empresa depois abusou e colocou seus próprios vôos com maior prioridade na tela de resultados das pesquisas. Assim sendo, eram escolhidos com maior frequência, interferindo no mecanismo de busca do Sabre. A empresa chamou isso de "Ciência Informativa". O Governo Norte Americano chamou de "Informação Polarizada e Deturpada" e banuiu essa prática do mercado.

Sabre foi um exemplo de duas vias: estratégia comercial e abuso comercial. Com o advento da Internet, busca por vôos

reapareceriam como o serviço Travelocity e sistemas de busca de propósito geral implementariam pagamento de quotas para apresentar posições privilegiadas como resultados de busca. Os Três Primeiros

Então como fica meu ranking de candidatos em uma lista de 1-12? Em ordem decrescente, os maiores software já escritos são:

12 - A Peste (Worm) Morris

11 - Sistema de Ranking de Busca do Google

10 - Sistema de Orientação Espacial da Apollo 11

9 - Planilha Eletrônica Excel

8 - Sistema Operacional Macintosh

7 - Sistema Sabre

6 - Navegador Mosaic

5 - Linguagem Java

4 - Sistema Operacional 360 da IBM Isso significa que a minha terceira, segunda e primeiríssima escolhas ainda estão faltando. Então vamos lá: O número três é o software de sequenciamento genético do Instituto de Pesquisa do Genoma. Não é um sistema cujos resultados são inexatos, e sim "uma guinada de brilhantismo técnico que consegue precisão de 10 em cada 10 sequências analisadas", comenta Morgenthaler. O sistema de sequenciamento do Instituto ajudou a subdividir e propiciar análises sequenciais, acrescido de sua habilidade de recombinar subunidades de análises como parte da análise integral, "acelerou a ciência genética em no mínimo uma década". Agora temos as ferramentas para traçar padrões da migração humana para a África; o genoma humano relevou com minúcias como a genética se diferencia entre grupos étnicos, exatamente em um momento onde essa informação é solenemente necessária. Ele concede base científica sobre como os humanos podem se olhar mutuamente como irmãos em um momento da história onde vivenciamos o risco de destruímos uns aos outros. O sistema vem sendo utilizado para realizar diversos sequenciamentos genéticos adicionais, mapeamento da raiz de várias doenças e montagem do quebra-cabeças da hereditabilidade que sempre assolou a Biologia. Pode pela primeira vez ser resolvido. Raras ocasiões aconteceram onde grandes pesquisas e grandes softwares estiveram tão ligados. Minha escolha número dois é o System R da IBM, um projeto de pesquisa da IBM no Laboratório Almaden em São Jose, Califórnia, que fez os bancos de dados relacionais ter seu rumo à ascensão. Na década de 70, Edgar Codd considerou a matemática teórica conjunta e concebeu uma maneira de aplicar essa estratégia no armazenamento de dados e obtenção desses dados de volta, usando a teoria dos conjuntos, que mantém elementos relacionados sem armazená-los em um compartimento identificado. Conjuntos são elementos relacionados que juntos desempenham um papel abstrato. O conjunto de cores azul, branco e vermelho por exemplo, são elementos relacionados que combinados dão origem às cores da bandeira da França, e que portanto dada a essa combinação geram um quarto elemento que pode posteriormente ser combinado à outros. É ainda possível encontrar todos os elementos de um único conjunto de forma improvisada, sabendo seguramente apenas um único elemento identificador desse conjunto. O System R e todos os que originaram-se desse princípio -- DB2, Oracle, Microsoft SQL Server, Sybase, PostgreSQL, MySQL, e tantos outros -- terão um impacto que nós apenas começamos a vislumbrar. Bancos de dados relacionais podem armazenar conjuntos de dados sobre clientes, e pesquisar outros conjuntos de dados para traçar o perfil de como um cliente faz compras. Os dados são adicionados na base na proporção com que são conseguidos, e o banco encontra relacionamentos escondidos nessas informações. As bases de dados relacionais e sua linguagem de acesso, a SQL, nos permite fazer algo que a mente humana quase considera impossível: encontrar uma quantia enorme de dados que se relacionam sem precisar saber muito do conteúdo desses dados, quando foi armazenado ou como se relacionava à outras informações. Tudo que é preciso é uma trecho adicional de informação apenas, a chave primária, que nos permite acessar o conjunto dos elementos. Gosto do System R por sua incrível suavidade operacional, sua escalabilidade e sua incrível utilidade para todos que trabalham com dados massivos. É um software com uma rara noção de verdade matemática por trás. E agora, O Maior Software Já Escrito pelo Homem -- Unix Os Laboratórios Bell frequentemente levam o crédito pela criação do sistema operacional Unix, mas Bell nunca financiou seu desenvolvimento. De fato, a gerência da instituição em questão não sabia nada sobre o assunto. Os Laboratórios Bell como parte da AT&T, tinham desenvolvedores comprometidos em um projeto de múltiplos fabricantes, chamado Multics, que fazia uso de muitas idéias novas para um sistema operacional. Mas o projeto sucumbiu, e um desses desenvolvedores da Bell Labs, Ken Thompson, decidiu fazer uma versão pessoal do Multics, de forma que pudesse concluir seus jogos de guerra nas estrelas, diz Feldman (que a propósito, foi o sétimo desenvolvedor do Unix da AT&T, e agora é presidente da Associação de Maquinários Computacionais). Na melhor das tradições de software, Unix foi um esforço individual que criou vida própria. Thompson projetou o Unix diante da rejeição dos Laboratórios Bell em dar continuidade ao projeto em uma estação PDP7 da DEC com 16 ou 32 Kbytes de memória -- Feldman não se lembra ao certo da capacidade de memória. "Unix foi escrito sob grandes limitações", comenta Feldman. "Não havia memória nem poder de processamento. Você se envergonharia hoje, se seu relógio de pulso por exemplo, não tivesse mais memória ou CPU do que o Unix dispunha naquele tempo". Thompson projetou seu pequeno sistema operacional de forma a mover dados em blocos ou em páginas, de um sistema de acesso aleatório de memória para um sistema de armazenamento secundário em disco, liberando mais espaço em memória principal. Quando essa memória fosse necessária novamente, o sistema operacional saberia acessar esses dados no disco e carregar de volta em memória aleatória. Desta forma, um grande sistema operacional pode rodar em um pequeno computador com pouca memória. Seu sistema operacional era ainda multiusuário. Até os mainframes daquela época eram limitados a usuários únicos, tornando o tempo computacional relativamente caro. O Sistema Operacional de Informações Uniplexadas (Unics) de Thompson, permitiria que duas pessoas utilizassem o computador ao mesmo tempo. O Grupo de Pesquisas de Ciências Computacionais dos Laboratórios Bell, ficaram sabendo do Unics e quiseram uma cópia. Sob requisição do grupo,

Thompson e seu parceiro Dennis Ritchie concordaram em adicionar um sistema de formatação de texto no ambiente operacional, com tanto que uma máquina maior, PDP 11/20 fosse fornecida. Então, o processamento de texto em Unix nasceu. Unics tornou-se Unix, foi reescrito em um sistema mais poderoso, mais portátil, em código de alto nível - em linguagem C - e foi anunciado mercadologicamente como o Unix System III da AT&T. Então o Unix System III foi o maior software já criado -- ou quase? Acompanhem comigo.

Uma Filosofia GNU

System III representou um avanço, mas esqueceu de muitas coisas como interface mais amigável com o usuário e um método de tratar sistemas distribuídos. Em uma tentativa de expandir a adoção do Unix, a AT&T o tornou disponível para instituições acadêmicas e de pesquisa sob uma pequena taxa. Algumas pessoas pensam que o Open Source apareceu quando os códigos tornaram-se livremente disponíveis pela Internet. Estão errados. De fato, o Código Aberto tem suas raízes na primeira distribuição do Unix. Uma dessas distribuições originou-se do trabalho noturno para melhorar o Unix na Universidade da Califórnia em Berkeley. Uma série de outros pesquisadores ouviram falar dessas melhorias, e pediram à pessoas como Bill Joy uma cópia desse Unix de Berkeley. Assim sendo, o primeiro código aberto não era um arquivo digital. Foi de fato, um rolo de fita magnética que Joy colocou no correio em uma dessas noites após terminar seu dia de trabalho, de acordo com Eric Allman, graduando da Universidade da Califórnia em Berkeley, criador do Sendmail, que estudou com Bill Joy. Em 1977, uma coleção das adições e melhorias de Joy e diversos outros alunos de graduação foi disposta em conjunto, e tornou-se conhecida como a Berkeley Software Distribution of the Unix Operating System, ou apenas a Distribuição de Software Unix de Berkeley -- o BSD. Unix foi projetado como módulos discretos de código, cada qual relacionado a um trecho específico do equipamento. Isso o tornava mais simples de ser analisado do que os sistemas operacionais da IBM. Os estudantes de Berkeley fizeram rápidas mudanças. Adicionaram um sistema de arquivos mais limpo e rápido e um subsistema de redes confiável, além do poderoso editor de textos e de códigos, o vi. Adicionaram a API de sockets de Berkeley, tornando a transmissão de dados para um destino remoto na rede tão simples quanto para um disco local. O Departamento de Defesa e Pesquisa Norte Americano havia contratado a consultoria Bolt Beranek & Newman para implementar um protocolo de rede para o governo. No BSD 4.1, os estudantes de Berkeley revisaram os conceitos por trás desse protocolo, modificaram e o aprimoraram, criando uma versão apropriada para atender suas próprias demandas. Em 1986, no BSD versão 4.3, o DARPA (Departamento de Defesa) testou esse protocolo, o TCP/IP, e decidiram adotá-lo ao invés da versão projetada por BBN. Bill Joy saiu da Universidade de Berkeley em 1982 para co-fundar a Sun Microsystems, usando Berkeley Software Distribution como base para o SunOS e Solaris. Sun e AT&T colaboraram mutuamente na melhoria do System V, produzindo o consolidado o Unix System V Release 4. Concordaram que esse seria o Unix padrão para o futuro. A AT&T, objetivando o retorno de seus investimentos no Unix, aumentou as taxas cobradas pelo software.

Mas os estudantes de Berkeley não saíram dos trilhos. Reescreveram o BSD Unix, livrando-o dos arquivos originais do Unix da AT&T e criando uma nova variante do sistema que poderia rodar em sistemas de baixo custo -- os computadores pessoais. Com mais uma versão do Unix, AT&T considerou que o impacto em seus lucros poderia ser comprometido, e sua nova subsidiária, a USL - Unix System Laboratories - processou a BSDI, empresa que distribuía uma versão do Unix de Berkeley destinada a computadores pessoais de linhagem Intel. A USL manteve tal processo por anos, atrasando a evolução do BSD Unix em ambiente computacional de linhagem Intel. Enquanto isso, as altas taxas cobradas pela licença do Unix da AT&T ofendiam Richard Stallman, aluno de graduação que utilizava o sistema no laboratório de inteligência artificial do MIT. Software, decidiu Stallman, deveria ser considerado recurso intelectual e ser livre, como o trabalho publicado de seus companheiros pesquisadores. Ele decidiu criar um conjunto de ferramentas com objetivo de substituir as ferramentas existentes no Unix, e que também fossem usadas por programadores para dar origem a outras ferramentas igualmente livres, e chamou-as de GNU. Essas ferramentas chegaram a Linus Torvalds, um estudante de 21 anos em uma universidade em Helsinki na Finlândia, exatamente quando este estava considerando uma versão do Unix que pudesse ser rodado em seu computador pessoal. Estas ferramentas foram decisivas na criação de um kernel de sistema operacional, que batizou com o nome Linux. O resto da história é conhecida. Linux tornou-se tão popular que ofuscou a Distribuição de Software de Berkeley em computadores pessoais. Hoje, Linux se esforça para assumir também a posição de mercado final. Mas Linux é meramente um clone de um sistema GNU incompleto e de seus predecessores BSD. BSD gerou todos os conceitos chave implementados no Linux; exatamente por isso todos os principais pilares da Internet como BIND, Sendmail e o TCP/IP foram desenvolvidos sob o Unix de Berkeley, e não no System V. É por isso que a Microsoft, ao precisar de uma implementação melhor do TCP/IP para Windows, utilizou a implementação BSD Unix. Quando a DARPA quis construir a rede ARPANET - hoje a Internet - em 1983, descartou seu protocolo existente e confiou o serviço no TCP/IP do BSD Unix. Então ai está: O Maior Software de Todos os Tempos, com o mais amplo impacto no mundo todo foi o BSD 4.3. Outros Unix tornaram-se sucesso comercial mais bem sucedido, mas as conquistas cumulativas conseguidas nos sistemas BSD, especialmente versão 4.3 representaram picos incomparáveis de inovação. BSD 4.3 representa sozinho a maior revolução da Internet. Mais que isso, a paixão que envolve o Linux e código aberto é apenas uma ramificação direta dos ideais criados no BSD: amor pelo poder computacional e a crença que isto deve ser uma extensão do poder intelectual humano, disponível sob todos os preceitos de liberdade -- poder de intelecto este que coloca o homem em seu devido lugar no universo.

A publicação original desse artigo está disponível aqui.